ARDC

AD

TECHNICAL REPORT NO. 8

THE BRLESC II INSTRUCTION CODE

by

Glenn A. Beck

February 1971

U.S. ARMY MATERIEL COMMAND
ABERDEEN RESEARCH AND DEVELOPMENT CENTER
ABERDEEN PROVING GROUND, MARYLAND

54

A B E R D E E N R E S E A R C H A N D D E V E L O P M E N T C E N T E R

TECHNICAL REPORT NO. 8

FEBRUARY 1971

The BRLESC II Instruction Code

Glenn A. Beck

Computer Support Division

Funded by all ARDC RDT&E Projects

A B E R D E E N P R O V I N G G R O U N D, M A R Y L A N D

A B E R D E E N R E S E A R C H A N D D E V E L O P M E N T C E N T E R

TECHNICAL REPORT NO. 8

GABeck/aji
Aberdeen Proving Ground, Md.
February 1971

The BRLESC II Instruction Code

ABSTRACT

This report describes the action and gives the execution times and instruction types for the 115 instructions of BRLESC II.

Preceding page blank

TABLE OF CONTENTS

		Page
	ABSTRACT	3
I.	INTRODUCTION	7
II.	MEMORY	8
	1. INSTRUCTIONS	8
	2. NUMBERS	9
III.	REGISTERS	12
	1. ARITHMETIC REGISTERS	12
	2. OTHER REGISTERS	12
IV.	CONTROL	13
V.	THE INSTRUCTION SET	14
	1. INTEGER AND FIXED POINT INSTRUCTIONS	14
	INTEGER	15
	EITHER INTEGER OR FIXED POINT	16
	FIXED POINT	19
	2. FLOATING POINT INSTRUCTIONS	20
	3. LOGICAL INSTRUCTIONS	24
	4. CONTROL INSTRUCTIONS	29
	5. SHIFT INSTRUCTIONS	32
	6. INDEX INSTRUCTIONS	35
	7. INPUT/OUTPUT INSTRUCTIONS	35
	CARD READING	38
	CARD PUNCHING	39
	PRINTER	39
	TAPE	40
	TAPE READ	41
	TAPE WRITE	41
	TAPE MOVE	42
	DISC	43
	8. UNUSED INSTRUCTION TYPES	45
VI.	SUMMARY AND TIMING OF INSTRUCTIONS	45
	APPENDIX A = ARDC PRINTER CHARACTERS	51
	DISTRIBUTION LIST	53

I. INTRODUCTION

BRLESC II is a large, high speed, electronic computer that is now in operation at ARDC. It was built to supplement the BRLESC I computer which has been operating since 1960.

This report is intended to aid programmers in writing assembly language programs for applications which cannot be done using the FORTRAN language and, in some cases, to aid in determining the cause when FORTRAN programs fail to execute as expected by the programmer. A description of each of the 115 executable instructions is given from a programming point of view. That is, a description is given of what is done, not how it is done.

II. MEMORY

The core memory is identical to and interchangeable with the BRLESC I memory. The memory can be manually switched from one machine to the other in banks of 16384 words. Each word is made up of 68 binary bits.

$b_{68}, b_{67}, \text{-------}, b_2, b_1$.

The 68 bits are usually considered as 17 sexadecimal digits, each representing 4 binary bits. The 16 possible values of a sexadecimal digit, in increasing sequence, are 0,1,2,3,4,4,5,6,7,8,9,K,S,N,J,F,L.

The memory access time is approximately 1.1 microseconds except for the first 256 words which can be either approximately 1.1 microseconds or approximately .2 microseconds depending on a switch setting on the machine.

1. Instructions

Any instruction can be either short (16 bits) or long (32 bits). The instruction type is stored in the left most 8 bits of the 16 or 32 bits and the associated address is stored in the right most 8 or 20 bits. Long instructions have 4 unused bits between the instruction type and the address. One machine word can store 2, 3, or 4 instructions. A short instruction can start in bit position b_{64}, b_{48}, b_{32}, and b_{16}. A long instruction can start in bit position b_{64}, b_{48}, and b_{32}. All 32 bits of a long instruction must be stored in the same instruction word.

Instructions do not use b_{67}-b_{65}. If b_{68} is one, then all instructions in the instruction word are ignored and control passes to the next instruction word.

Instructions must be marked short or long. This is done in the rightmost bit of the 8 bit instruction type. If the bit is zero, the instruction is long.

2 Numbers

All numbers except exponents of real numbers are stored using 2's complement representation. To change the sign of a number, all of the bits are inverted (zeros changed to ones and ones to zeros) and then one is added in the rightmost bit position of the result. There are 3 kinds of numbers (and 3 kinds of instructions) namely, fixed point, integer and floating point. Integer values have an implied binary point following b_1. Fixed point and floating point values have an implied binary point between b_{61} and b_{60}. A sexadecimal exponent is stored in $b_8 - b_1$ in floating point values. This exponent is biased by 128. Thus the values of numbers are as follows:

$$\text{Integer value} = -2^{64}\, b_{65} + \sum_{i=1}^{64} b_i 2^{i-1}$$

$$\text{Fixed point value} = -16\, b_{65} + \sum_{i=1}^{64} b_i 2^{i-61}$$

$$\text{Floating point value} = \left[-16b_{65} + \sum_{i=9}^{64} b_i 2^{i-61}\right]\left[16 \sum_{i=1}^{8} b_i\, 2^{8i-1} - 128\right]$$

EXAMPLES OF NUMBERS

Decimal value	Sexadecimal value
integer 1	00000000000000001
integer -1	1LLLLLLLLLLLLLLL
integer 50	00000000000000032
integer -100	1LLLLLLLLLLLLL9N
Fixed point 1.25	01400000000000000
Fixed point 1.1 (actually $1.1+.4.2^{-60}$)	0119999999999999K
Fixed point - .0625	1LL00000000000000
Floating point 2.	02000000000000080
Floating point 32.	02000000000000081
Floating point -1.	1000000000000007L
Floating point -.125	1F000000000000007L

The word formats are as follows:

INTEGER FORMAT

3 bits	1 bit	64 bits	.
unused	sign	integer value	binary point

FIXED POINT FORMAT

3 bits	1 bit	4 bits	.	60 bits
unused	sign	Integer part of value	binary point	Fractional part of value

FLOATING POINT FORMAT

3 bits	1 bit	4 bits	•	52 bits	8 bits
unused	sign	Integer part of coeffecient	binary point	Fractional part of coeffecient	biased exponent

III. REGISTERS

1. Arithmetic registers

There are three 68 bit registers, A, R and D, which are program accessable. The result of nearly all of the instructions is stored in the A register. The R register can be generally considered as a continuation of the A register. The D register is used as a temporary storage register for the execution of many of the instructions.

2. Other registers

The index register. I

There is one index register which can be set by any of four instructions. The contents of the index register is available to modify by addition the address of the next instruction which is not a SKIP instruction. This result is the effective address of the instruction (EA). After the index register has been used to modify an address, it will have no effect until it has been set again.

The exponent register EX.

There is one exponent register which contains the exponent of the last executed floating point operation. This can be the exponent of A or R, whichever was defined last. During floating point operations the exponents of the values are not stored in he A or R register but in EX instead. The last 8 bits of A and R can contain bits of the results of floating point operations.

The instruction register IR.

This register contains the instructions which came from one machine word and contains the instruction presently being executed. There is no instruction which accesses this register.

The next instruction register NI

This register contains the address of the word from which the next instruction will come after the instructions which are presently in the instruction register have been executed. This register is increased by one when the contents of the word specified in it are transferred to the instruction register. It is also changed by the unconditional, conditional, and secondary jumps and by the count-down clock.

The past jump register PJ

This register contains the address of the last jump instruction increased by one.

The past past jump register PPJ

This register contains the address of the next to last jump instruction increased by one.

Clocks

The machine has two clocks. One contains the present time, giving the month, day, hour, minute and hundredths of minute. The other is an integer count-down clock. The integer is reduced by 1 every .01 minute. The machine executes a jump to memory word 058 (sexadecimal) when the count reaches zero.

IV. CONTROL

Instructions are executed left to right thru a word and thru successive words when there is no transfer of control by an unconditional jump, conditional jump or secondary jump instruction. If b_{68} of an instruction word is one, then instructions in that word are not executed and control is passed to the following word. All jump instructions transfer control to the left half or right half of an instruction (not to the 2nd or 4th quarter).

A secondary jump differs from the unconditional and conditional jumps in 2 ways. First, any instruction remaining in the instruction word are executed before the jump takes place. Second, if there is an unconditional jump or a conditional jump which jumps or another secondary jump in the remaining instructions of the word, then the secondary jump has no effect.

V. THE INSTRUCTION SET

The instruction set is broken into 7 catagories: fixed point and integer, floating point, logical, control, shift, index, and input-output.

The first line of the description of each instruction gives the symbolic type, acceptable by the FORTRAN and FORAST compilers, followed by the sexadecimal type. The sexadecimal type given is for a long instruction; add 1 for the corresponding short instruction. The order type is followed by a description of the operation.

1. Fixed point and integer instructions.

When a value is transferred from the memory to one of the registers A, D or R by an integer or fixed point instruction, the 4 left most bits of the register (b_{68} - b_{65}) are set to the same value as b_{65} of the memory word. When a value is transferred from the A register to a memory word by any of the integer or fixed point instructions, bits 68-66 of the memory word are set to zero and bits 65-1 of the A register are stored in bits 65-1 of the memory word.

The fixed point and integer instructions can be broken into 3 classes. First, those which perform integer operations (a binary point is assumed to the right of b_1.) Second, those which perform fixed point (a binary point is assumed between b_{61} and b_{60}) or integer operations. Third, those which perform fixed point operations.

Integer instructions

The following 9 instruction types are usually used in integer operations. The binary point is assumed following b_1.

+1	02	One is subtracted from the contents of the effective address and the result is stored in the A register. One is stored in the D register.
EA(-)	0F	The effective address is subtracted from the contents of the A register and the result is stored in the A register. The effective address is stored in the D register.
EA-	32	The effective address is subtracted from zero and the result is stored in the A register. The effective address is stored in the D register.
EA(+)	36	The effective address is added to the contents of the A register. The result is stored in the A register. The effective address is stored in the D register.
EA+	K2	The effective address is stored in the A register.
1M	S0	One is stored in the effective address and in the A register.
1(+)M	S2	The contents of the effective address are increased by 1 and stored in the effective address and in the A register. The original contents of the effective address are stored in the D register.
IX	S8	The contents of the A register are multiplied by the contents of the effective address and the result is stored in the A register. The original contents of both the R register and the D register are destroyed during the operation.

EAX	NF	The contents of the A register are multiplied by the effective address and the result is stored in the A register. The original contents of both the R register and the D register are destroyed during the operation.

Either Integer or Fixed Point Instructions.

The following 22 instruction types can be used as fixed point or integer instructions. There is no assumption made by the hardware as to the position of the binary point.

(-)	04	The contents of the effective address are subtracted from the contents of the A register and the result is stored in the A register. The contents of the effective address are stored in the D register.
A(-)	ON	The contents of the R register are subtracted from the contents of the A register and the result is stored in the A register. The contents of the R register are stored in the D register. If the effective address is not zero, a secondary jump to the effective address is executed.
M	10	Bits 68-66 of the effective address are set to zero and bits 65-1 of the effective address are set to bits 65-1 of the A register.
-	24	The contents of the effective address are subtracted from zero and the result is stored in the A register. The contents of the effective address are stored in the D register.
A-	2N	The contents of the R register are subtracted from zero and the result is stored in the A register. The contents of the R register are stored in the D register. If the effective address is not zero, a secondary jump to the effective address is executed.

-HV	44	The absolute value of the contents of the effective address is subtracted from the contents of the A register and the result is stored in the A register. The contents of the effective address are stored in the D register.
A+M	4N	The contents of the R register are stored in the A register; then a symbolic type M (sexadecimal type 10) instruction is executed.
M-A	54	The contents of the A register are subtracted from the contents of the effective address and the result is stored in the A register. The original contents of the A register are stored in the D register.
-A	58	The contents of the A register are subtracted from zero and the result is stored in the A register. The original contents of the A register are stored in the D register. If the effective address is not zero, a secondary jump to the effective address is executed.
1-1	64	The absolute value of the contents of the effective address is subtracted from zero and the result is stored in the A register. The contents of the effective address are stored in the D register.
A1-1	6N	The absolute value of the contents of the R register is subtracted from zero and the result is stored in the A register. The contents of the R register are stored in the D register. If the effective address is not zero, a secondary jump to the effective address is executed.
+HV	84	The absolute value of the contents of the effective address is added to the contents of the A register and the result is stored in the A register. The contents of the effective address are stored in the D register.

'A'	8N	The contents of the D register are stored in the A register. If the effective address is not zero, a secondary jump to the effective address is executed.
(+)M	98	The contents of the effective address are added to the contents of the A register and the result is stored in the A register and in the effective address. The original contents of the effective address are stored in the D register.
+	K4	The contents of the effective address are stored in the A register.
A+	KN	The contents of the R register are stored in the A register and the D register. If the effective address is not zero, a secondary jump to the effective address is executed.
R	S4	The contents of the effective address are stored in the R register.
(+)	N4	The contents of the effective address are added to the contents of the A register and the result is stored in the A register. The contents of the effective address are stored in the D register.
A(+)	NN	The contents of the R register are added to the contents of the A register and the result is stored in the A register. The contents of the R register are stored in the D register. If the effective address is not zero, a secondary jump to the effective address is executed.
(-)M	J0	The contents of the effective address are subtracted from the contents of the A register and the result is stored in the A register and in the effective address. The original contents of the effective address are stored in the D register.

1+1	F4	The absolute value of the contents of the effective address is stored in the A register. The contents of the effective address are stored in the D register.
A1+1	FN	The absolute value of the contents of the R register is stored in the A register. The contents of the R register are stored in the D register. If the effective address is not zero, a secondary jump to the effective address is executed.

Fixed Point Instructions

The binary point is assumed between b_{61} and b_{60} in each of the 5 following fixed point instructions.

SQRT	5N	The square root of the contents of the A register is stored in the A register and the D register. The contents of the R register are destroyed during this operation. If the effective address is not zero, a secondary jump to the effective address is executed.
/	78	The contents of the A register are divided by the contents of the effective address and the result is stored in the A register. The result is also stored in the R register but the sign is not spread into bits 68-66 of the R register. The remainder, multiplied by 2^n, is stored in the D register, where n is the number of consecutive bits b_{64}---b_{64-n} of the denominator which are different than b_{65}.
/A	7N	The contents of the effective address are divided by the contents of the A register and the result is stored in the A register. The result is also stored in the R register but the sign is not spread into bits 68-66 of R register. The remainder, multiplied by 2^n,

is stored in the D register, where n is the number of consecutive bits b_{64}--b_{64-n} of the denominator which are different than b_{65}.

XU K8 The contents of the A register are multiplied by the contents of the effective address and the most significant part of the result (68 bits) is stored in the A register and the least significant part of the result (60 bits) is stored in the R register. Bits 68-61 of the R register are set to the sign of the product. The original contents of the A register are stored in the D register.

XA J8 The contents of the A register are multiplied by the contents of the effective address and the most significant part of the result (68 bits) is rounded and stored in the A register. The least significant part of the result is stored in the R register. Bits 68-61 of the R register are set to the sign of the product. The original contents of the A register are stored in the D register.

2. Floating point instructions

When values are transferred to any of the registers from the memory, the sign bit (bit 65) is also stored in b_{68}-b_{66} of the register. The exponent portion of the register, b_8-b_1, is set to zero. If the value is transferred to the A register or R register, the exponent is stored in the EX register. The floating add and subtract operations store the contents of the effective address in the D register. Then, if the exponents are not equal, the coefficient of the smaller of A and D is shifted right to align the sexadecimal point. If the A register is shifted, the R register is also shifted.

After each floating operation, the coefficient of the result (contents of the A register) is checked. If it is not zero and outside of both of the ranges $-16 \leq C < -1/16$ and $1/16 \leq C < 16$ then it is shifted so that it does fall into one of the ranges and the exponent is adjusted accordingly. If a shift is required, then the contents of the R register are also shifted.

F(-) 06 The contents of the effective address are subtracted from the contents of the A and EX registers and the result is stored in the A and EX registers.

FM 12 The coefficient stored in the A register is checked. If it is not zero and outside of both of the ranges $-16 \leq C < -1$ and $1 \leq C < 16$, then it is shifted so that it does fall into one of the ranges and the exponent is adjusted accordingly. If a shift is required, the contents of the R register are also shifted. The result in the A register is then rounded by adding one to b_9 if $b_8 = 1$. This addition may cause the coefficient to be outside the above ranges. In this case, another shift and exponent adjustment would be made. Then $b_{65} - b_9$ of the A register are stored in $b_{65} - b_9$ of the effective address and the contents of the EX register are stored in $b_8 - b_1$ of the effective address. Zeros are stored in b_8-b_1, of the A register. Bits 68-65 of the effective address are set to zero.

F- 26 The contents of the effective address are subtracted from zero and stored in the A and EX registers.

F-HV 46 The absolute value of the contents of the effective address is subtracted from the A and EX registers and stored in the A and EX registers.

FM-A 56 The contents of the A and EX registers are subtracted from the contents of the effective address and the result is stored in the A and EX registers.

F-A	5K	The contents of the A and EX registers are subtracted from zero and the result is stored in the A and EX registers. If the effective address is not zero, a secondary jump to the effective address is executed. The original contents of the A register are stored in the D register.
FSQRT	5F	The square root of the contents of the A and EX registers is stored in the A and EX registers. The coefficient of the result is also stored in the D register. The contents of the R register are destroyed during the operation. If the effective address is not zero, a secondary jump to the effective address is executed.
F1-1	66	The absolute value of the contents of the effective address is subtracted from zero and the result is stored in the A and EX registers. The contents of the effective address are stored in the D register with the exponent portion set to zero.
F/	7K	The contents of the A and EX registers are divided by the contents of the effective address and the result is stored in the A and EX registers. The coefficient of the result is also stored in the R register but the sign is not spread into $b_{68} - b_{65}$ of the R register. The contents of the D register are destroyed by the operation.
F/A	7F	The contents of the effective address are divided by the contents of the A and EX registers and the result is stored in the A and EX registers. The coefficient of the result is also stored in the R register but the sign is not spread into $b_{68} - b_{65}$ of the R register.

		The contents of the D register are destroyed by the operation.
F+HV	86	The absolute value of the contents of the effective address is added to the contents of the A and EX registers and the result is stored in the A and EX registers.
F(+)M	9K	The following two instructions are executed: (1) Symbolic type F(+) (sexadecimal type N6) (2) Symbolic type FM (sexadecimal type 12)
F+	K6	The contents of the effective address are stored in the A and EX registers.
FR	S6	The contents of the effective address are stored in the R and EX registers.
F(+)	N6	The contents of the effective address are added to the contents of the A and EX registers and the result is stored in the A and EX registers.
F(-)M	J2	The following two instructions are executed: (1) Symbolic type F(-) (Sexadecimal type 06) (2) Symbolic type FM (Sexadecimal type 12)
FXA	JK	The contents of the A and EX registers are multiplied by the contents of the effective address and the result is stored in the A and EX registers. The original contents of the A register are stored in the D register. The result is continued in the R register starting at bit 68.
F1+1	F6	The absolute value of the contents of the effective address is stored in the A and EX registers.

3. Logical instructions

The logical instructions operate with all 68 bits of a machine word on a bit by bit basis. The extract operations replace 20 bits of a left or right address.

There are four logical operations defined as follows:

(1)	Inclusive or	IOR	0 IOR 0 = 0
			0 IOR 1 = 1
			1 IOR 0 = 1
			1 IOR 1 = 1
(2)	Exclusive or	EOR	0 EOR 0 = 0
			0 EOR 1 = 1
			1 EOR 0 = 1
			1 EOR 1 = 0
(3)	and	AND	0 AND 0 = 0
			0 AND 1 = 0
			1 AND 0 = 0
			1 AND 1 = 1
(4)	Not	NOT	NOT 0 = 1
			NOT 1 = 0

IORM	2K	The following two instructions are executed: (1) Symbolic type IOR (Sexadecimal type 8K) (2) Symbolic type LM (Sexadecimal type 48)
OM	30	Store zero in all of the bits of the A register and the effective address. The EX register is not changed.
LM	48	The contents of the A register are stored in the effective address.

ANDM	3K	The following two instructions are executed: (1) Symbolic type AND (Sexadecimal type FK) (2) Symbolic type LM (Sexadecimal type 48)
A+LM	4K	The contents of the R register are stored in the A register and in the effective address.
E	50	The rightmost 20 bits (b_{20} - b_1) of the A register are stored in the rightmost 20 bits of the effective address.
PJE'	52	The contents of the PJ register are stored in the rightmost 20 bits of the effective address and in the rightmost 20 bits of the D register. The remaining bits of the D register are set to zero.
0E'	70	Store zeros in the A register and in the last 20 bits of the effective address.
PPJE'	72	The contents of the PPJ register are stored in the rightmost 20 bits of the effective address and in the rightmost 20 bits of the D register. The remaining bits of the D register are set to zero.
EOR	88	The exclusive or operation is executed with the contents of the A register and the contents of the effective address. The 68 corresponding bits of the two operands are combined to form a 68 bit result which is stored in the A register. The contents of the effective address are stored in the D register.

IOR	8K	The inclusive or operation is executed with the contents of the A register and the contents of the effective address. The 68 corresponding bits of the two operands are combined to form a 68 bit result which is stored in the A register. The contents of the effective address are stored in the D register.
RPE'	8F	The integer i-n is stored in the D register, where n is the value of the effective address of the last executed RPR (sexa. 96) or RPB (sexa. 92) instruction and i is the number of times the first instruction of the instruction word last repeated was executed. The rightmost 20 bits of the integer i-n are also stored in the rightmost 20 bits of the effective address.
E	90	The 20 bits b_{52}-b_{33} of the A register are stored in bits b_{52}-b_{33} of the effective address.
RSW	94	The 17 sexadecimal digits set on the manual read switches on the console are stored in the A register. If the effective address is not zero, a secondary jump to the effective address is executed.
RCLK	9N	The present value of the real time clock is stored into the rightmost 60 bits of the effective address. The time is recorded as ten 6 bit characters, two each for month, day, hour, minute and hundredths of minute. Bits 64-61 of the effective address are set to zero. The number of 16384 word memory

		banks available is stored in the left most sexadecimal character of the effective address (b_{68} - b_{65}).
SMAXT	9F	The effective address is stored in the count-down clock. The value is then reduced by 1 every .01 min and the machine is interrupted when the count reaches zero. This instruction should not be used except by the operating system.
L+	KK	The contents of the effective address are stored in the A register.
LR	SK	The contents of the effective address are stored in the R register.
NOT	N2	The NOT operation is executed with the 68 bits of the contents of the effective address and the result is stored in the A register. The contents of the effective address are stored in the D register.
ANDN	NK	The NOT operation is executed with the contents of the effective address; then the AND operation is executed with this result and the contents of the A register and the result is stored in the A register. The contents of the effective address are stored in the D register.
NOP	F2	This instruction does nothing except use the index register to compute an effective address and set the index register to zero.
	N8	This instruction sets the R register the same as the RER instruction (sexa.F8). The condition indicators are not cleared.

RER F8 The following table indicates the bits of the R register which are set to one if the corresponding condition has occurred and to zero if the condition has not occurred. The remaining bits of R are set to zero.

bit	condition
9	Divide by zero (fixed or floating).
10	Divide exceed (fixed or floating).
11	Square root of negative number (fixed or floating).
12	Floating exponent overflow.
13	Exceed memory or I/O memory request interlock.
	All of the above conditions cause the machine to stop.
18	Disc parity error.
	The following two conditions will be added later:
22	A tape read has passed over a file mark.
23	Memory parity error.

If the effective address is not zero, a secondary jump to the effective address is executed. The condition indicators are cleared.

AND FK The AND operation is executed with the contents of the A register and the contents of the effective address and the result is stored in the A register. The contents of the effective address are stored in the D register.

SKIP	FF	This instruction does nothing. If the index register has been set, it will remain set until the execution of an instruction that is not a SKIP instruction.
0E	LO	Store zeros in the A register and in $b_{52} - b_{33}$ of the effective address.

4. Control Instructions

Most of the following instructions are jump instructions. When an instruction which causes a jump is executed, the following registers are changed:

(1) The contents of the PJ register are stored in the PPJ register.
(2) The contents of the NI register are stored in the PJ register.
(3) The effective address of the instruction is stored in the NI register.

U'	14	Jump to the instruction on the right side of the effective address.
C	20	If bit 65 of the A register is 0, jump to the instruction on the left side of the effective address.
C-	22	If bit 65 of the A register is 1, jump to the instruction on the left side of the effective address.
0U'	34	Store zeros in the A register and jump to the instruction on the right side of the effective address.

C'	40	If bit 65 of the A register is 0, jump to the instruction on the right side of the effective address.
C'-	42	If bit 65 of the A register is 1, jump to the instruction on the right side of the effective address.
CZ	60	If the 68 bits of the A register are zero, jump to the instruction on the left side of the effective address.
CNZ	62	If any of the 68 bits of the A register are not zero, jump to the instruction on the left side of the effective address.
CZ'	80	If all 68 bits of the A register are zero, jump to the instruction on the right side of the effective address.
CNZ'	82	If any of the 68 bits of the A register are one, jump to the instruction on the right side of the effective address.
RPB	92	Repeat the instruction in the next word that does not contain an instruction that sets the index register the number of times indicated by the effective address. All of the addresses are incremented after each execution. The increment is one if the RPB instruction is not followed by an instruction which sets the index register. If there is an instruction which sets the index register following the RPB instruction, then the value of the index register is the increment.

		Any of the addresses may be either short or long, but if an address is short it must not be incremented enough to become long.
RPR	96	This instruction is the same as RPB except that only the instructions that have their order type in the right half of the word are incremented.
OU	K0	Store zeros in the A register and jump to the instruction on the left side of the effective address.
U*	SN	Jump to the instruction on the left side of the effective address. This instruction is usually used only to jump to subroutines.
EXC	SF	The contents of the effective address are stored in the instruction register and executed next. If there is no jump executed, control returns to the word following the word which contained the EXC instruction. The instructions (if any) following the EXC instruction in the same word are not executed. The NI, PJ and PPJ registers are not changed by the EXC instruction. The effective address of any EXC instruction may contain another EXC instruction. This can happen any number of times but control will return to the word following the first EXC instruction if none of the other executed instructions jumped.
U	N0	Jump to the instruction on the left side of the effective address.

COV	JN	Jump to the left side of the effective address if the left most 4 bits (b_{68}-b_{65}) of the A register do not have the same value. If the contents of the A register is the result of a fixed point, add, subtract, or multiply and the left most 4 bits do not have the same value, then there has been a fixed point overflow.
COV'	JF	Same as COV except jump to the right side of the effective address.
ZX	F0	This instruction causes the computer to stop or continue if the ZX switch is on or off respectively. If the switch is on, then the machine will continue if the initiate button is pressed. In any case, a secondary jump is executed if the effective address is not zero.
HALT	LN	This instruction causes the machine to stop. If any I/O operations are being executed at the time of the HALT, they will be completed. The machine will continue with the next instruction if the initiate button is pressed.

5. Shift Instructions

All 7 of the shift instructions shift part of or all of the contents of the A and R registers. The number of binary places shifted is specified by the rightmost 8 bits of the value of the effective address. A shift of zero places is permitted. The descriptions below are given in terms of repeated shifts of one. The machine has the following shift paths built into the hardware; right 1,2,3,4,5,6,8,12, 14, and 16 and left 1,2,4,6 and 8. These are used in various combinations to minimize the execution time.

RS	08	The following operations are executed the number of times specified by the value of the effective address. Bits 64-1 of the A register and bits 60-1 of the R register are shifted right one place. The bit 1 shifted out of the A register is shifted to bit 60 of the R register. The bit 1 shifted out of the R register is lost. Bit 64 of the A register is replaced with bit 65 of the A register. Bits 68-65 of the A register and 68-61 of the R register are not changed.
RSL	0K	The following operations are executed the number of times specified by the value of the effective address. Bits 68-1 of the A register and bits 68-1 of the R register are shifted right one place. The bit 1 shifted out of the A register is shifted to bit 68 of the R register. The bit 1 shifted out of the R register is lost. Bit 68 of the A register is set to zero.
LS	18	The following operations are executed the number of times specified by the value of the effective address. Bits 64-1 of the A register and bits 68-1 of the R register are shifted left one place. The bit 64 shifted out of the A register is shifted to bit 1 of the R register. The bit 68 shifted out of the R register is lost. Bit 1 of the A register is set to zero. Bits 68-64 of the A register are not changed.

LSD	1K	The following operations are executed the number of times specified by the value of the effective address. Bits 68-1 of A register and bits 60-1 of the R register are shifted left one place. The bit 68 shifted out of the A register is shifted to bit 1 of the R register. The bit 60 shifted out of the R register is shifted to bit 1 of the A register. Bits 68-61 of the R register are not changed.
LSC	1N	The following operations are executed the number of times specified by the value of the effective address. Bits 68-1 of the A register and bits 68-1 of the R register are shifted left one place. The bit 68 shifted out of the A register is shifted to bit 1 of the R register. The bit 68 shifted out of the R register is shifted to bit 1 of the A register.
RSC	28	The following operations are executed the number of times specified by the value of the effective address. Bits 68-1 of the A register and bits 68-1 of the R register are shifted right one place. The bit 1 shifted out of the A register is shifted to bit 68 of the R register. The bit 1 shifted out of the R register is shifted to bit 68 of the A register.
LS0	38	Clear the A register to zero and then do LSD (sexadecimal type 1K) instruction.

6. Index Instructions

The 4 index instructions set the index register which will be added to the address of the next instruction which is not a SKIP to form the effective address. If an index instruction follows one of the repeat instructions, then the value stored in the index register is used as the address increment during the execution of the repeat.

MI1	16	The integer one is subtracted from the contents of the A register and the rightmost 20 bits of the result are stored in the index register.
I1	1F	The integer one is subtracted from the contents of the effective address and the rightmost 20 bits of the result are stored in the index register.
I	3N	The rightmost 20 bits of the contents of the effective address are stored in the index register.
EAI	3F	The effective address is stored in the index register.

7. Input/Output Instructions

The four input output instructions are:

Symbolic	Sexa.
IOS	L2
SDA	L8
SIA	L4
SFA	L6

The first (IOS) selects the equipment to be used, the direction of the transfer (input or output), the number of bits per character, the number of bits per word, and the parity (even or odd). The second (SDA) is used only in disc I/O. It selects the position on the disc to be used. The third (SIA) and fourth (SFA) specify how much information is to be transferred and the area in the memory to be used.

The instructions must be executed in the order given in the above list. Any number of instructions may be executed between the execution of any of them. No I/O action takes place until the execution of an SFA instruction.

The machine has four input/output channels. All four channels can operate concurrently. The machine also executes other instructions concurrently with any I/O operation except the machine will stop if any reference is made by a non-I/O instruction to any word included in any I/O instruction and will continue on after the I/O instruction has completed.

The effective address of the IOS instruction is broken into 5 sexadecimal characters $C_5\ C_4\ C_3\ C_2\ C_1$. Each one has a special meaning and will be briefly discussed separately.

If $C_2 = 8$	Read cards if $C_1 = 0$
	Print on printer if $C_1 = 1$
	Punch cards if $C_1 = 2$
$C_2 = 4$	Use disc
$C_2 = 0$	Use the tape unit specified by C_1.

The individual bits of C_3, C_4 and C_5 have the following meaning:

C_3 bit 1	= 0	60 bit wds (64 bit wds when C_3 bit 2 = 1)
	= 1	72 bit wds
C_3 bit 2	= 0	6 bit char.
	= 1	8 bit char. (12 bit char. when reading cards)
C_3 bit 3	= 0	Read
	= 1	Write
C_3 bit 4	= 0	Odd parity
	= 1	even parity
C4 bit 1	= 0	foreward (tape only)
	= 1	backward
C4 bit 2	= 0	blocks (tape only)
	= 1	file marks
C4 bit 3	= 0	not move (tape or disc)
	= 1	move
C4 bit 4	= 0	not rewind (tape only)
	= 1	rewind
C5 bit 1	= 0	not unload tape
	= 1	unload tape (unload means automatically remove tape from read head, tape cannot be used without operator intervention)
C5 bit 2	= 0	use parity
	= 1	ignore parity

Input-Output instructions start reading or writing at the effective address of the SIA instruction. The number of words read or written is the effective address of the SFA instruction minus the effective address of the SIA instruction. Exceptions to this rule are explained below.

The 60 (and 64) bit reads and writes use the rightmost 60 (and 64) bits of the word. The 72 bit writes includes the 4 parity bits of the memory word. When reading, bits 68-61 for 60 bit read and 68-65 for 64 bit read are set to zero.

Card reading

IOS (080) or IOS (CARD)

Read cards 6 bits per column and store 10 characters (60 bits) per word. The card code for each 6 bit character and corresponding bit code is listed in appendix A.

IOS (0380) or IOS (CARD-R72-C8)

Read cards 12 bits per column and store 6 characters (72 bits) per word. The first 4 bits of the first character of each word are lost. The 12 bits are defined by the 12 rows in a card column. A nonpunched row reads as zero and a punched row reads as one. The 12 bits from top to bottom of the card define the 12 bits from left to right. Not more than 9 words (78 columns) should be read with this instruction.

IOS (0280) or IOS (CARD-C8)

Read cards 12 bits per column and store 5 of these 12 bit characters (60 bits) per word.

IOS (0180) or IOS (CARD-R72)

Read cards 6 bits per cclumn and store 12 characters (72 bits) per word. The first 4 bits of the first character of each word are lost. Not more than 6 words (72 columns) should be read with this instruction.

Card Punching

IOS(0682) or IOS(CARD-C8-W60) to select left card hopper
IOS (040682) to select middle card hopper
IOS (080682) to select right card hopper

Cards are punched by rows. The bits are recorded on the rows in the same order as they appear in the machine words. Sixty four bits are taken from each word. Thus, b_{64}-b_1 of the first word and b_{64}-b_{49} of the second word are recorded in the first (top) row of the card (b_{64} is recorded in column 1). The remaining bits, b_{48}-b_1, of the second word and b_{64}-b_{33} of the third word are recorded in the second row. This process continues taking 5 words for each 4 rows on the card. Fifteen words are required to record a complete card. The middle and right card hoppers are used by the operating system.

Printer

IOS (0481) or IOS (PRINT)

The printer has two options determined by a printer console switch. The first is 80 columns per line. The contents of 8 words are printed on a line. The information comes from the rightmost 60 bits of a word. Each word defines ten six bit characters.

The second option is variable length line or formatted print. The maximum line length is 132 columns. Ten 6 bit characters are taken from each memory word. The 6 bit character 111111 is an ignore character and is not printed (it does not take a column position on the printed page). The first character of a line that is not an ignore character determines which channel of a paper tape is to be selected. The paper tape then determines how far the paper is advanced before the line is printed. The first character is not printed. The end of a line character is 011111. It is not printed.

Tape Instructions

A block on a tape is the data recorded by a tape write instruction. If a tape read instruction requires less data than is recorded in the block, then the tape is moved to the end of the block. If a tape read instruction requires more data than is recorded in the block, then the read stops at the end of the block. The remaining memory words specified by the read instruction are not changed. If the number of characters in the tape block is not an integer multiple of the number of characters stored in a word, then the last partial word is left adjusted and the right end filled with binary ones. All tapes are 1/2 inch wide and 2400 feet long. The tape units have 7 track read-write heads or 9-track read-write heads. The instructions which write and read 6 bit character data can use either a 7 or 9 track unit but must use the same for reading as was used for writing. The instructions which write and read 8 bit character data must use a 9 track unit. Presently, there is only one 7 track unit on the machine.

In all tape instructions described below, u is a decimal integer indicating a tape switch number ($1 \leq u \leq 15$) and X is a sexadecimal character indicating a tape switch number ($1 \leq X \leq L$).

TAPE READ INSTRUCTIONS

IOS (0X) or IOS (R-u) for odd parity
or IOS (080X) or IOS (R-EP-u) for even parity
or IOS (02000X) or IOS(R-IP-u) for ignore parity

Read 6 bit characters and store 10 of the characters (60 bits) per word.

IOS (010X) or IOS (R72-u) for odd parity
or IOS (090X) or IOS (R72-EP-u) for even parity
or IOS (02010X) or IOS (R72-IP-u) for ignore parity.

Read 6 bit characters and store 12 of these characters (72 bits) per word.

IOS (020X) or IOS (R-C8-u) for odd parity
or IOS (0K0X) or IOS (R-C8-EP-u) for even parity
or IOS (02020X) or IOS (R-C8-IP-u) for ignore parity

Read 8 bit characters and store 8 of these characters (64 bits) per word.

IOS (030X) or IOS (R72-C8-u) for odd parity
or IOS (0S0X) or IOS (R72-C8-EP-u) for even parity
or IOS (02030X) or (R72-C8-IP-u) for ignore parity

Read 8 bit characters and store 9 of these characters (72 bits) per word.

TAPE WRITE INSTRUCTIONS

IOS (040X) or IOS (W-u) for odd parity
or IOS (0N0X) or IOS (W-EP-u) for even parity.
or IOS (02040X) or IOS (W-IP-u) for ignore parity.

Write 6 bit characters taking 10 characters (60 bits) from each word.

IOS (050X) or IOS (W72-u) for odd parity.
or IOS (0J0X) or IOS (W72-EP-u) for even parity.
or IOS (02050X) or IOS (W72-IP-u) for ignore parity.

Write 6 bit characters taking 12 characters (72 bits) from each word.

IOS (060X) or IOS (W-C8-u) for odd parity.
or IOS (0F0X) or IOS (W-C8-EP-u) for even parity.
or IOS (02060X) or IOS (W-C8-IP-u) for ignore parity.

Write 8 bit characters taking 8 characters (64 bits) from each word.

IOS (070X) or IOS (W72-C8-u) for odd parity.
or IOS (0L0X) or IOS (W72-C8-EP-u) for even parity.
or IOS (02070X) or IOS (W72-C8-IP-u) for ignore parity.

Write 8 bit characters taking 9 characters (72 bits) from each word.

TAPE MOVE INSTRUCTIONS

In the following 4 move instructions, N is the difference between the effective address of the SFA instruction and the effective address of the SIA instruction.

IOS(0400X) or IOS(MF-u)
Move the tape specified forward N blocks

IOS(0500X) or IOS(MB-u)
Move the tape specified backward N blocks

IOS(0600X) or IOS(MFMF-u)

Move the tape specified forward N file marks

IOS(0700X) or IOS (MFMB-u)

Move the tape specified backward N file marks

IOS(0900X) or IOS(REW-u)

Move the tape specified backward to the beginning of the tape. If the tape is already rewound the instruction does nothing. The addresses of the SIA and SFA instructions are not used but the instructions must be executed.

IOS(01900X) or IOS (UNLOAD-u)

This is the same as the previous instruction except that the tape is removed from the read head and the unit must be set up by the operator before another tape instruction calling for the same unit is executed.

IOS(0240X) or IOS(WFM-u)

Write a file mark on the unit specified. The addresses of the SIA and SFA instructions are not used but the instructions must be executed.

DISC INSTRUCTIONS

The disc is made of 10 surfaces with 203 cylinders on each surface for a total of 2030 tracks. Each track can store approximately 400 72 bit words or 450 64 bit words. A disc read or write starts only at the beginning of a track. A disc write must not specify more data than can be recorded on a track. The disc does not start recording on the next track when a track becomes full.

The 10 disc surfaces are addressed 0 thru 9 and the 203 cylinders are addressed 0 thru 202 in decimal or 0 thru ONK in sexadecimal. The disc will hang up and must be reset manually if:

(1) A surface address is greater than 9,

(2) A cylinder address is greater than 202,

(3) An attempt is made to write on the portion of the disc which is locked out for writing. Normally this portion is cylinders 0-127 (0-7L in sexadecimal).

IOS(04040) or IOS(DISC-MF)

Move the disc read head to the surface specified by the last 4 bits of the effective address of the SIA instruction and the cylinder specified by the last 8 bits of the effective address of the SFA instruction.

The IOS instruction may be followed by an SDA instruction (sexadecimal type L8) in all of the following disc instructions. If the SDA instruction does not appear, the disc will use the track specified last. If the SDA instruction does appear, the disc head is moved before the read or write as follows: The last 4 bits of the effective address specify the surface; the next 8 bits (bits 12-5) specify the cylinder. Only 8 bit characters are recorded on and read from the disc.

IOS(0240) or IOS(DISC-R-C8)

Read from the disc 8 bit characters and store 8 characters (64 bits) per word.

IOS(0340) or IOS(DISC-R72-C8)

Read from the disc 8 bit characters and store 9 characters (72 bits) per word.

IOS(0640) or IOS (DISC-W-C8)

Write on the disc 8 bit characters taking 8 characters (64 bits) from each word.

IOS(0740) or IOS (DISC-W72-C8)

Write on the disc 8 bit characters taking 9 characters (72 bits) from each word.

8. Unused instruction types

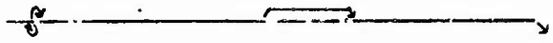
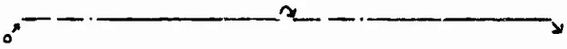
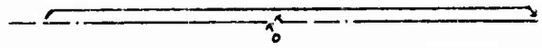
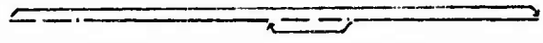
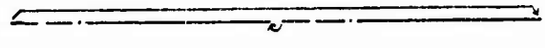
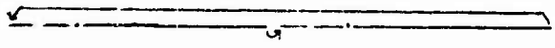
The following sexadecimal types are not used. The machine will stop if an attempt is made to execute any of them.

C0, 2F, 4F, 68, 6K, 6F, 74, 76, KF, J4, J6, LK, and LF.

IV. SUMMARY AND TIMING OF INSTRUCTIONS

Key:				
	M	Memory contents	R	R register contents
	A	A register contents	D	D register contents
	EA	Effective address		
	F	Leading F on symbolic type indicates fl. pt. operation		
	I	Index register for indexing next instruction		
	v	Inclusive OR logical operator		
	^	AND logical operator		
	-	NOT logical operator		
	1	Integer one (means absolute value in symbolic types)		

The times given are average execution times in microseconds. They do include the memory accesses for bringing operands from the memory and storing the results in the large memory. If the small memory is being used, subtract .7 microseconds for each small memory access. The times do not include the access time to bring the instruction to the instruction register. For total execution time, add 1/2 microsecond for each instruction for instructions stored 2 per word, 1/3 microsecond for instructions stored 3 per word and 1/4 microsecond for instructions stored 4 per word. The average is probably about 1/3. The sexadecimal order type is for a long instruction, add one for a short instruction.

Sexa.	Sym	Time	Page	Action
00			45	
02	+1	1.45	15	A=M-1
04	(-)	1.45	16	A=A-M
06	F(-)	1.6	21	A=A-M
08	RS	.5	33	
0K	RSL	.5	33	
0N	A(-)	.34	16	A=A-R Secondary jump
0F	EA(-)	.40	15	A=A-EA
10	M	1.35	16	M=A(65 bits)
12	FM	1.5	21	M=A
14	U'	.5	29	Jump to right order
16	MI1	1.05	35	I=A-1
18	LS	.6	33	
1K	LSD	.6	34	
1N	LSC	.6	34	
1F	I1	1.9	35	I=M-1
20	C	.5	29	Jump to left order if A ≥ 0
22	C-	.5	29	Jump to left order if A < 0
24	-	1.45	16	A=-M
26	F-	1.45	21	A=-M
28	RSC	.5	34	
2K	IORM	2.5	24	M=A=A v M
2N	A-	.34	16	A=-R Secondary jump
2F			45	
30	0M	1.35	24	M=A=0
32	EA-	.40	15	A=-EA
34	0U'	.5	29	A=0 and jump to right order

Sexa.	Sym	Time	Page	Action
36	EA(+)	.4	15	A=A+EA
38	LSo	.6	34	A=0, then execute LSD instruction
3K	ANDM	2.8	25	M=A=A $\wedge$ M
3N	I	1.6	35	I=M
3F	EAI	.5	35	I=EA
40	C'	.5	30	Jump to right instruction of EA if A$\geq$0
42	C'-	.5	30	Jump to right instruction of EA if A<0
44	-HV	1.45	17	A=A - |M|
46	F-HV	2.	21	A=A - |M|
48	LM	1.35		M=A(68 bits)
4K	A+LM	1.45	25	M=A=R(68 bits)
4N	A+M	1.45	17	M=A=R(65 bits)
4F			45	
50	E'	2.3	25	b_{20}-b_1 of M= b_{20}-b_1 of A
52	PJE'	2.4	25	b_{20}-b_1 of M= Past jump address
54	M-A	1.55	17	A=M-A
56	FM-A	2.	21	A=M-A
58	-A	.5	17	A=-A. Secondary jump
5K	F-A	.6	22	A=-A. Secondary jump
5N	SQRT	48.	19	A= A. Secondary jump
5F	FSQRT	48.	22	A= A. Secondary jump
60	CZ	.5	30	Jump to left instruction if A=0
62	CNZ	.5	30	Jump to left instruction if A $\neq$ 0
64	1-1	1.45	17	A= - |M|
66	F1-1	1.45	22	A= - |M|
68			45	
6K			45	
6N	A1-1	.34	17	A= - |R| Secondary jump

Sexa.	Sym	Time	Page	Action
6F			45	
70	0E'	2.3	25	A=0, then do E' instruction
72	PPJE'	2.4	25	b_{20}-b_1 of M= Past past jump address
74			45	
76			45	
78	/	22.8	19	A=A/M
7K	F/	23.	22	A=A/M
7N	/A	22.9	19	A=M/A
7F	F/A	23.1	22	A=M/A
80	CZ'	.5	30	Jump to right instruction if A=0
82	CNZ'	.5	30	Jump to right instruction if A≠0
84	+HV	1.45	17	A= |M|
86	F+HV	1.45	23	A= |M|
88	EOR	1.45	25	A=A^MvA^M (Exclusive or)
8K	IOR	1.45	26	A=AvM
8N	'A'	.34	18	A=D Secondary jump
8F	RPE'	2.4	26	b_{20}-b_1 of M= repeat count i-n
90	E	2.3	26	b_{52}-b_{33} of M=b_{52}-b_{33} of A
92	RPB	.38	30	Repeat, advance all addresses.
94	RSW	.3	26	A= Switches. Secondary jump
96	RPR	.38	31	Repeat, advance right addresses.
98	(+)M	2.8	18	M=A=A+M
9K	F(+)M	3.	23	M=A=A+M
9N	RCLK	1.45	26	M=clock.
9F	SMAXT		27	set count-down clock.
K0	0U	.5	31	A=0, then do U instruction.
K2	EA+	.4	15	A=EA
K4	+	1.45	18	A=M (65 bits)

Sexa	Sym	Time	Page	Action
K6	F+	1.45	23	A=M
K8	XU	14.4	20	A,R= A*M
KK	L+	1.4	27	A=M(68 bits)
KN	A+	.34	18	A=R Secondary jump
KF			45	
S0	1M	1.45	15	M=A=1
S2	1(+)M	2.8	15	M=A=M+1
S4	R	1.45	18	R=M(65 bits)
S6	FR	1.45	23	R=M
S8	IX	5.	15	A=A*M(integers)
SK	LR	1.45	27	R=M (68 bits)
SN	U*	.5	31	Jump to subroutine.
SF	EXC	1.6	31	Do the instructions at EA
N0	U	.5	31	Jump to left instruction.
N2	NOT	1.45	27	A =$\overline{M}$
N4	(+)	1.45	18	A=A+M
N6	F(+)	2.	23	A=A+M
N8		.3	27	R=Error bits. Indicators not cleared.
NK	ANDN	1.45	27	A=A$\wedge\overline{M}$
NN	A(+)	.34	18	A=A+R Secondary jump
NF	EAX	3.	16	A=A*EA (integers)
J0	(-)M	2.8	18	M=A=A-M
J2	F(-)M	3.	23	M=A=A-M
J4			45	
J6			45	
J8	XA	14.4	20	A=A*M
JK	FXA	13.4	23	A=A*M
JN	COV	.5	32	Jump to left if fixed pt. overflow.
JF	COV'	.5	32	Jump to right if fixed pt. overflow.
F0	ZX		32	Conditional Halt. Secondary jump
F2	NOP	.34	27	No operation. I is used.

F4	1+1	1.45	19	A= |M|
F6	F1+1	1.45	23	A= |M|
F8	RER	.3	28	R= Error bits. Indicators cleared.
FK	AND	1.45	28	A=A∧M
FN	A1+1	.34	19	A= |R| Secondary jump
FF	SKIP	.02	29	No operation. I is not used
L0	0E	2.3	29	A=0, then do E instruction
L2	IOS		35	I/O select
L4	SIA		35	I/O initial address
L6	SFA		35	I/O final address +1
L8	SDA		35	Set disc address
LK			45	
LN	HALT		32	Stop
LF			45	

APPENDIX A. ARDC PRINTER CHARACTERS

DEC. EQUIV.	CARD CODE	BIT CODE	CHAR.	DEC. EQUIV.	CARD CODE	BIT CODE	CHAR.
0	blank	00 0000	blank	32	11	10 0000	-
1	1	00 0001	1	33	11-1	10 0001	J
2	2	00 0010	2	34	11-2	10 0010	K
3	3	00 0011	3	35	11-3	10 0011	L
4	4	00 0100	4	36	11-4	10 0100	M
5	5	00 0101	5	37	11-5	10 0101	N
6	6	00 0110	6	38	11-6	10 0110	O
7	7	00 0111	7	39	11-7	10 0111	P
8	8	00 1000	8	40	11-8	10 1000	Q
9	9	00 1001	9	41	11-9	10 1001	R
10	2-8	00 1010	&	42	11-2-8	10 1010	!
11	3-8	00 1011	=	43	11-3-8	10 1011	$
12	4-8	00 1100	'	44	11-4-8	10 1100	*
13	5-8	00 1101	:	45	11-5-8	10 1101	]
14	6-8	00 1110	>	46	11-6-8	10 1110	;
15	7-8	00 1111	"	47	11-7-8	10 1111	↑
16	12	01 0000	+	48	0	11 0000	0
17	12-1	01 0001	A	49	0-1	11 0001	/
18	12-2	01 0010	B	50	0-2	11 0010	S
19	12-3	01 0011	C	51	0-3	11 0011	T
20	12-4	01 0100	D	52	0-4	11 0100	U
21	12-5	01 0101	E	53	0-5	11 0101	V
22	12-6	01 0110	F	54	0-6	11 0110	W
23	12-7	01 0111	G	55	0-7	11 0111	X
24	12-8	01 1000	H	56	0-8	11 1000	Y
25	12-9	01 1001	I	57	0-9	11 1001	Z
26	12-2-8	01 1010	?	58	0-2-8	11 1010	←
27	12-3-8	01 1011	.	59	0-3-8	11 1011	,
28	12-4-8	01 1100	)	60	0-4-8	11 1100	(
29	12-5-8	01 1101	[	61	0-5-8	11 1101	%
30	12-6-8	01 1110	<	62	0-6-8	11 1110	\
31	12-7-8	01 1111	#	63	0-7-8	11 1111	@

Bit codes 01 1111(#) and 11 1111 (@) are used as end-of-line and ignore characters respectively for variable length lines.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) U. S. Army Aberdeen Research and Development Center Aberdeen Proving Ground, Maryland 21005	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

THE BRLESC II INSTRUCTION CODE

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

Glenn A. Beck

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February 1971	53	0

8a. CONTRACT OR GRANT NO. b. PROJECT NO. Funded by all ARDC RDT&E Projects c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ARDC Technical Report No. 8 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY U. S. Army Materiel Command Washington, D. C. 20315

13. ABSTRACT

This report describes the action and gives the execution times and instruction types for the 115 instructions of BRLESC II.

DD FORM 1 NOV 65 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Digital Computer Instruction Code Assembly language BRLESC II Computer						